Development and Evaluation of an Open-Ended Computational Evolution System for the Genetic Analysis of Susceptibility to Common Human Diseases

Jason H. Moore, Peter C. Andrews, Nate Barney, and Bill C. White

Computational Genetics Labroatory, Department of Genetics
Dartmouth Medical School, Lebanon, NH, USA
{Jason.H.Moore,Peter.C.Andrews,Nate.Barney,Bill.C.White}@dartmouth.edu

Abstract. An important goal of human genetics is to identify DNA sequence variations that are predictive of susceptibility to common human diseases. This is a classification problem with data consisting of discrete attributes and a binary outcome. A variety of different machine learning methods based on artificial evolution have been developed and applied to modeling the relationship between genotype and phenotype. While artificial evolution approaches show promise, they are far from perfect and are only loosely based on real biological and evolutionary processes. It has recently been suggested that a new paradigm is needed where "artificial evolution" is transformed to "computational evolution" (CE) by incorporating more biological and evolutionary complexity into existing algorithms. It has been proposed that CE systems will be more likely to solve problems of interest to biologists and biomedical researchers. The goal of the present study was to develop and evaluate a prototype CE system for the analysis of human genetics data. We describe here this new open-ended CE system and provide initial results from a simulation study that suggests more complex operators result in better solutions.

1 Introduction

1.1 The Problem Domain: Human Genetics

Human genetics is undergoing an information explosion and an understanding implosion. This is the result of technical advances that make it feasible and economical to measure 10^6 or more DNA sequence variations from across the human genome. For the purposes of this paper we will focus exclusively on the single nucleotide polymorphism or SNP which is a single nucleotide or point in the DNA sequence that differs among people. Most SNPs have two alleles (e.g. *A* or *G*) that combine in the diploid human genome in one of three possible genotypes (e.g. *AA*, *AG*, *GG*). It is anticipated that at least one SNP occurs approximately every 100 nucleotides across the $3x10^9$ nucleotide human genome. An important goal in human genetics is to determine which of the many hundreds of thousands of SNPs are useful for predicting who is at risk for common diseases. Further, it is important to know the nature of the mapping elationship between genotypes at the important SNPs and the phenotype or clinical endpoint. This knowledge is useful for identifying those at risk and for informing experimental studies that can lead to new therapeutic interventions.

The charge for computer science and bioinformatics is to develop algorithms for the detection and characterization of those SNPs that are predictive of human health and disease. Success in this genome-wide endeavor will be difficult due to nonlinearity in the genotype-to-phenotype mapping relationship that is due, in part, to epistasis or nonadditive gene-gene interactions. Epistasis was recognized by Bateson [1] nearly 100 years ago as playing an important role in the mapping between genotype and phenotype. Today, this idea prevails and epistasis is believed to be a ubiquitous component of the genetic architecture of common human diseases [2]. As a result, the identification of genes with genotypes that confer an increased susceptibility to a common disease will require a research strategy that embraces, rather than ignores, this complexity [2,3,4]. The implication of epistasis from a data mining point of view is that SNPs need to be considered jointly in learning algorithms rather than individually. Because the mapping between the attributes and class is nonlinear, the concept difficulty is high. The challenge of modeling attribute interactions has been previously described [5]. The goal of the present study is to develop an evolutionary computing strategy for detecting and characterizing epistasis.

1.2 A Simple Example of the Concept Difficulty

Epistasis can be defined as biological or statistical [3]. Biological epistasis occurs at the cellular level when two or more biomolecules physically interact. In contrast, statistical epistasis occurs at the population level and is characterized by deviation from additivity in a linear mathematical model. Consider the following simple example of statistical epistasis in the form of a penetrance function. Penetrance is simply the probability (P) of disease (D) given a particular combination of genotypes (G) that was inherited (i.e. $P[D|G]$). A single genotype is determined by one allele (i.e. a specific DNA sequence state) inherited from the mother and one allele inherited from the father. For most single nucleotide polymorphisms or SNPs, only two alleles (encoded by A or a) exist in the biological population. Therefore, because the order of the alleles is unimportant, a genotype can have one of three values: AA, Aa or aa. The model illustrated in Table 1 is an extreme example of epistasis. Let's assume that genotypes AA, aa, BB, and bb have population frequencies of 0.25 while genotypes Aa and Bb have frequencies of 0.5 (values in parentheses in Table 1). What makes this model interesting is that disease risk is dependent on the particular combination of genotypes inherited. Individuals have a very high risk of disease if they inherit Aa or Bb but not both (i.e. the exclusive OR function). The penetrance for each individual genotype in this model is 0.5 and is computed by summing the products of the genotype frequencies and penetrance values. Thus, in this model there is no difference in disease risk for each single genotype as specified

Table 1. Penetrance values for genotypes from two SNPs

	AA (0.25)	Aa (0.50)	aa (0.25)
BB (0.25)	0	1	0
Bb (0.50)	1	0	1
bb (0.25)	0	1	0

by the single-genotype penetrance values. This model was first described by Li and Reich [6]. Heritability, or the size of the genetic effect, is a function of these penetrance values. In this model, the heritability is maximal at 1.0 because the probability of disease is completely determined by the genotypes at these two DNA sequence variations. As Freitas [5] reviews, this general class of problems has high concept difficulty.

1.3 Towards Computational Evolution for the Analysis of Gene-Gene Interactions

Numerous machine learning and data mining methods have been developed and applied to the detection of gene-gene interactions. These include, for example, traditional methods such as neural networks [7] and novel methods such as multifactor dimensionality reduction [8]. Evolutionary computing methods such as genetic programming (GP) have been applied to both attribute selection and model discovery in the domain of human genetics. For example, Ritchie et al. [8] used GP to optimize both the weights and the architecture of a neural network for modeling gene-gene interactions. More recently, GP has been successfully used for both attribute selection [9,10] and genetic model discovery [11].

Genetic programming is an automated computational discovery tool that is inspired by Darwinian evolution and natural selection [12,13,14,15,16,17,18]. The goal of GP is evolve computer programs to solve problems. This is accomplished by first generating random computer programs that are composed of the building blocks needed to solve or approximate a solution to a problem. Each randomly generated program is evaluated and the good programs are selected and recombined to form new computer programs. This process of selection and recombination is repeated until a best program is identified.

Genetic programming has been applied successfully to a wide range of different problems including data mining and knowledge discovery [e.g. [19]] and bioinformatics [e.g. [20]]. Despite the many successes, there are a large number of challenges that GP practitioners and theorists must address before this general computational discovery tool becomes one of several tools that a modern problem solver calls upon [21]. Banzhaf et al. [22] propose that overly simplistic and abstracted artificial evolution (AE) methods such as GP need to be transformed into computational evolution (CE) systems that more closely resemble the complexity of real biological and evolutionary systems. Evolution by natural selection solves problems by building complexity. As such, computational systems inspired by evolution should do the same. The working hypothesis addressed in the present study is that a GP-based genetic analysis system will find better solutions faster if it is implemented as a CE system that can evolve a variety of complex operators that in turn generate variability in solutions. This is in contrast to an AE system that uses a fixed set of operators.

1.4 Research Questions Addressed and Overview

The goal of the present study was to develop and evaluate an open-ended CE system for the detection and characterization of epistasis. We developed a hierarchically-organized and spatially-extended GP approach that is capable of evolving its own operators of any arbitrary size and complexity. The primary question addressed in this study is whether the ability to evolve complex operators improves the ability of the system to discover a classifier that is capable of predicting disease in the presence of nonlinear gene-gene interactions.

2 A Prototype Computational Evolution System

Our primary goal was to develop a prototype CE system that is capable of open-ended evolution for bioinformatics problem-solving in the domain of human genetics. Figure 1 gives a graphical overview of our hierarchically-organized and spatially-extended GP system that is capable of open-ended CE. At the bottom layer of this hierarchy is a grid of solutions. Details of the solutions and their representation are given in Section 2.1. At the second layer of the hierarchy is a grid of operators of any size and complexity that are capable of modifying the solutions. The operators are described in Section 2.2. At the third layer in the hierarchy is a grid of mutation operators that are capable of modifying the solution operators. The mutation operators are described in Section 2.3. At the highest level of the hierarchy is the mutation frequency that determines the rate at which operators are mutated. This is described in Section 2.4. Details of how the system was implemented are described in Section 2.5. The details of the experimental design used to evaluate this system are described in Section 3.

2.1 Problem Solutions: Their Representation, Fitness Evaluation and Reproduction

The goal of a classifier is to accept as input two or more discrete attributes (i.e. SNPs) and produce a discrete output that can be used to assign class (i.e. healthy or sick). Here, we used symbolic discriminant analysis or SDA as our classifier. The SDA method [23] has been described previously for this problem domain [11]. Briefly, SDA models consist of a set of attributes and constants as input and a set of mathematical functions that produce for each instance in the dataset a score called a symbolic discriminant score. The goal of SDA is to find a linear or nonlinear combination of attributes such that the difference between the distributions of symbolic discriminant scores for each class is maximized. Here, our SDA function set was $\{+, -, *, /, \%, <, <=, >, >=, ==, \neq\}$ where the % operator is a mod operation and / is a protected division. The SDA models are represented as postfix expressions here instead of as expression trees as has been used in the past [23,11] to facilitate stack-based evaluation of the classifiers and to facilitate representation in text files.

Classification of instances into one of the two classes requires a decision rule that is based on the symbolic discriminant score. Thus, for any given symbolic discriminant score (S_{ij}) in the ith class and for the jth instance, a decision rule can be formed such that if $S_{ij} > S_o$ then assign the instance to one class and if $S_{ij} <= S_o$ then assign the observation to the other class. When the prior probability that an instance belongs to one class is equal to the probability that it belongs to the other class, S_o can be defined as the arithmetic mean of the median symbolic discriminant scores from each of the two classes. This is the classification rule we used in the present study and is consistent with previous work in this domain [11]. Using this decision rule, the classification accuracy for a particular discriminant function can be estimated from the observed data. Here, accuracy is defined as $(TP+TN)/(TP+TN+FP+FN)$ where TP are true positives (TP), TN are true negatives, FP are false positives, and FN are

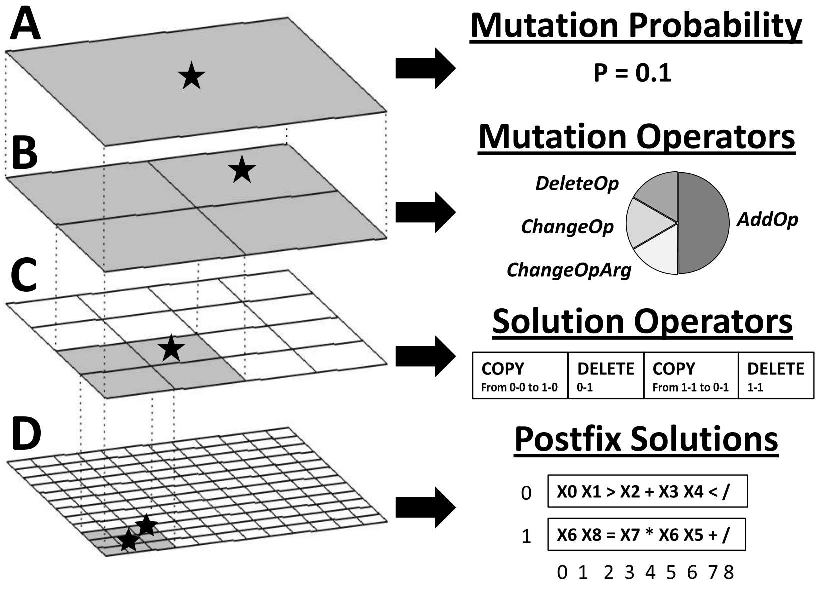

Fig. 1. Visual overview of our prototype CE system. The hierarchical structure is shown on the left while some specific examples at each level are shown on the right. The top two levels of the hierarchy (A and B) exist to generate variability in the operators that modify the solutions. Shown in C is an example set of operators that will perform recombination on the two solutions shown in D. As illustrated in B, there is a 0.50 probability that a mutation to the recombination operator in C will add an operator thus making this particular operator more complex. This system allows operators of any arbitrary complexity to modify solutions. Note that we used a 24x24 grid of solutions in the present study. A 12x12 grid is shown as an illustrative example.

false negatives. We used accuracy as the fitness measure for SDA solutions as has been described previously [11].

All SDA solutions in a population are organized on a toroidal grid with specific X and Y coordinates (see example in Figure 1). As such, they resemble previous work on cellular genetic programming [24]. In the present study we used a 24x24 grid for a total population size of 576. Reproduction of solutions in the population is handled in a spatial manner. Each solution is considered for reproduction in the context of its Moore neighborhood using an elitist strategy. That is, each solution in question will compete with its eight neighbors and be replaced in the next generation by the neighbor with the highest fitness of all solutions. This combines ideas of tournament selection that is common in GP with a set of solutions on a grid. Variability in solutions is generated using hierarchically organized operators. This is described below.

2.2 Operators for Computational Evolution: Generating Solution Variability

Traditional AE approaches such as GP use a fixed set of operators that include mutation and recombination, for example. The goal of developing a prototype CE system was to provide operators and building blocks for operators that could be combined to create new operators of any arbitrary complexity. We started with the following six operators and operator building blocks. The first operator, DeleteRangeOperation, deletes all functions in an SDA postfix expression within a certain range. The second operator, CopyRangeOperator, copies all functions in an SDA postfix expression within a certain range to another SDA postfix expression at a particular position. The third operator, PermuteRangeOperator, randomizes the order of a set of SDA functions within a given range. The fourth operator, AddOperator, adds a randomly selected function onto the end of a set of SDA functions. The fifth operator, PointMutationOperator, replaces a function and its arguments (e.g. attributes) at a given position with a randomly selected function and arguments. The final operator, PointMutationExpertKnowledgeOperator, replaces a function and its arguments (e.g. attributes) at a given position with a randomly selected function and arguments selected using a source of expert knowledge. Greene et al. [25] have shown that using ReliefF measures of attribute quality to guide point mutation for genetic analysis using GP is beneficial for ensuring good building blocks are utilized. This is consistent with Goldberg's ideas about exploiting good building blocks in competent genetic algorithms [26]. Thus, we have provided to the CE system a set of operators and operator building blocks that can be put together in any arbitrary length and complexity. For example, a standard recombination operator can be formed by combining two CopyRangeOperator operators and two DeleteRangeOperation operators with the appropriate arguments that specify the correct positions in two SDA solutions for copying and deleting appropriate model pieces. An example recombination operator is shown in Figure 1. These operators can be combined in more interesting ways to form even more complex operators.

As with the solutions, each operator is organized on a toroidal grid with a specific X and Y coordinate. Rather than generate one operator for each solution we assigned each operator to a set of solutions. This makes evaluation of the fitness of an operator easier since its positive or negative effect on the solutions can be averaged over multiple solutions. In this study, we assigned each operator to a 6x6 grid of 36 solutions. Thus, the population of operators is organized in a 4x4 grid for a total of 16 operators (See Figure 1) that each maps onto 36 of the 576 solutions.

2.3 Mutation of Operators for Computational Evolution: Generating Operator Variability

An important goal for the prototype CE system is the ability to generate variability in the operators that modify solutions. To accomplish this goal we developed an additional level in the hierarchy (Figure 1B) with mutation operators that specifically alter the operators described above. We defined four different fixed mutation operators that are each assigned to a 2x2 grid of solution operators. Solution operators can be modified in the following four ways. First, an operator can have a specific operator building block deleted (DeleteOperator). Second, an operator can have a specific operator building block added (AddOperator). Third, an operator can have a specific operator building block changed (ChangeOperator). Finally, an operator can have its arguments changed (ChangeOperatorArguments). This latter function allows, for example, the range that a DeleteRangeOperation would use. For our prototype, we fixed the probabilities with which each of these types of mutations can change the operators. Here, we used all four types of mutation and defined four different probability distributions for their use. For the first distribution we set the probabilities for DeleteOperator, AddOperator, ChangeOperator and ChangeOperatorArguments to 0.5, 0.167, 0.167 and 0.167 respectively. For the second distribution we set the probabilities to 0.167, 0.5, 0.167 and 0.167. For the third we set the probabilities to 0.167, 0.167, 0.5 and 0.167 and for the fourth we set the probabilities to 0.167, 0.167, 0.167 and 0.5. This preliminary assignment of probabilities allows us to explore the usefulness of each type of mutation. In future versions the type of mutation and their probabilities will also evolve. These four sets of mutations that alter solution operators exist in a 2x2 grid. Each mutates four sets of operators at the next level down in the hierarchy (see Figure 1).

2.4 Mutation Frequency

The top level of the CE system hierarchy (see Figure 1) is the mutation frequency that controls the probability that one of the four mutation sets in the next level down will mutate a given solution operator two levels down. In the present study we fixed this to 0.1. In future version this will be an evolvable parameter. Note that this frequency does not control the frequency with which an operator modifies a solution in the lowest level. This is controlled by the operator itself when it specifies which solution(s) it will modify.

2.5 Implementation

The CE system was programmed in C++. A single run of the system with a population of 576 solutions on a 24x24 grid for 100 generations took approximately three minutes on an 2.2 GHz AMD Opteron processor.

3 Experimental Design

Our goal was to provide an initial evaluation of the prototype CE system described above. The central question addressed in this study is whether the ability to evolve operators of any arbitrary complexity improves quality of the SDA models. To address this question, we first ran, as a baseline, the CE system that utilized only a simple mutation operator. Next, we ran the CE system with all available operators. Each run was completed with a population size of 576 (24x24 solutions) for 100 generations and 1000 generations. The best model from each run was saved along with the accuracy of the symbolic discriminant function. Each method was run 100 times with different random seeds on data that was simulated using the penetrance function in Table 2 below. The data consisted of 1600 instances and two functional SNPs that are associated with class only through the type of nonlinear interaction described in Section 1.2. The heritability of this model is 0.4. Each dataset also consisted of 98 randomly generated SNPs that represent potential false-positives or noise in the data. The challenge for the CE system is to search for the right combination of two SNPs and identify a nonlinear function that approximates the pattern generated by the penetrance model in Table 2. It is important to note that target classification accuracy for the correct model is approximately 0.8.

Table 2. Penetrance values for genotypes from two SNPs used to simulate data

	AA (0.04)	Aa (0.32)	aa (0.64)
BB (0.04)	0.486	0.960	0.538
Bb (0.32)	0.947	0.004	0.811
bb (0.64)	0.640	0.606	0.908

The distribution of accuracies obtained from running the CE system with just a simple mutation operator versus running the system with the capability of generating more complex operators were statistically compared using a two-sample t-test. The two systems were considered statistically significant at a type I error rate of 0.05.

4 Results

Figure 2 below summarizes the distribution of accuracies obtained from running the CE system 100 times on the simulated data with evolved operators (All)

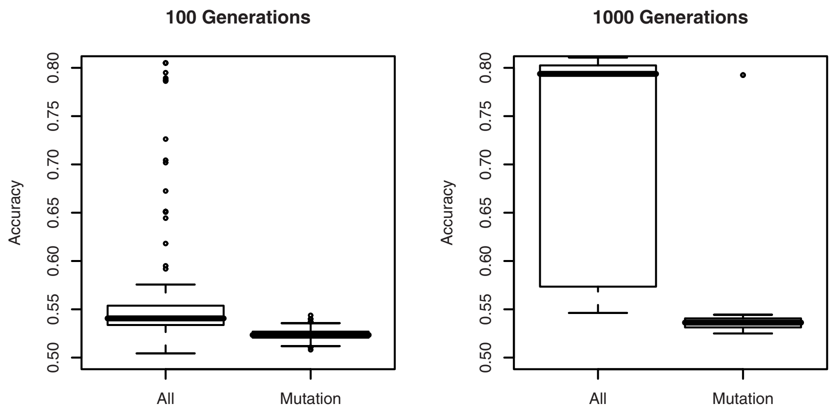

Fig. 2. Boxplots summarizing the distribution of accuracies obtained from running the CE system 100 times on the simulated data with evolved operators (All) or with just a mutation perator (Mutation) for 100 generations and 1000 generations

or with just a mutation operator (Mutation) for 100 generations and 1000 generations. The line in the middle of each box is the median of the distribution while the upper and lower limits of the box itself represent the 25th and 75th percentiles. The dashed lines extending from each box represent the approximate range of values with circles representing extreme values. Note that at 1000 generations, mutation alone only approximated the correct answer once out of 100 runs while the full CE system approximated the correct answer more than 50% of the time. In both cases, the mean accuracy was significantly higher for the full system ($P < 0.05$).

These preliminary results indicate that, for this specific domain, a CE system with the ability to evolve operators of any size and complexity does indeed identify better solutions than a baseline system that uses a fixed mutation operator. An important question is whether more complex operators were actually used to generate the best models discovered by the CE system. We evaluated the operators discovered during each run that were associated with a best model and found that all six operators and operator building blocks defined in Section 2.2 were used at least once in each of the 100 runs. This demonstrates that complex operators were discovered and used to generate better solutions than a simple mutation operator was able to generate.

5 Discussion and Conclusions

Banzhaf et al. [22] have suggested that traditional artificial evolution methods such as genetic programming (GP) will greatly benefit from our current understanding of the complexity of biological and evolutionary systems. They propose a new research agenda in which CE systems that mimic the complexity of biological systems will replace the overly simplified artificial evolution systems that have been inspired by biology, but largely ignore the complexity of biological processes. The goal of the present study was to specifically address whether a computational evolution system capable of evolving more complex operators will find better solutions than an artificial evolution system in the domain of human genetics. To accomplish this goal we developed a prototype CE system that is both spatially and hierarchically organized and is capable of evolving operators of any arbitrary size and complexity from a set of basic operator building blocks. Our preliminary experimental results demonstrate that the ability to evolve more complex operators does indeed improve the ability of the system to identify good models. These results support our working hypothesis and are consistent with the research agenda proposed by Banzhaf et al. [22].

It is important to note that the system presented here is a prototype and, as such, there are many extensions and modifications that can be made that would be consistent with CE. We first discuss several features implemented in the prototype that add complexity to the system and then propose some additional features inspired by the complexity of biological systems. There were two primary sources of complexity. First, the system is capable of evolving a diversity of different operators that modify solutions in the spatially-organized population. This is similar to real biological systems that evolve more complex genomic processes. For example, microRNAs that participate in post-translational regulation have evolved, in part, to help determine developmental processes such as body plan specification. Sempere et al. [27] showed that the number of microRNAs an animal group has correlates strongly with the hierarchy of metazoan relationships. The ability of species to evolve new biological processes plays an important role in increasing their complexity. As a second feature, we have included in the set of operator building blocks a mutation function that responds to the environment (i.e. the expert knowledge). We know that expert knowledge in the form of other data mining results or biological information about gene function is critical for success in this domain [9,10,11]. Here, we gave the CE operators the ability to use expert knowledge (i.e. information from the environment) in the form of pre-processed ReliefF scores to preferentially choose good attributes as arguments for a new function. The ability of an organism to respond to its environment plays an important role in fitness. The important role of environmental sensing has been discussed [22].

Our future goal is to improve the prototype CE system by adding additional features that are inspired by the complexity of real biological systems. As a first step, we will make the mutation operators (see Section 2.3, Figure 1B) more complex by giving them the ability to evolve. That is, the probability distribution that controls how operators are modified through mutation will evolve with feedback from how the system is doing. We also make the overall mutation frequency at the highest level an evolvable parameter. The evolvability of the entire system will make it attractive to implement this system in parallel as an island model thus providing a virtual ecosystem with feedback between populations. As a second step, we will add additional feedback loops in the system. For example, the solutions could contribute information back to the environment that is then used by a complex operator to generate variability in solutions. This takes the environmental sensing idea discussed above a step further. We anticipate the addition of these types of feedback loops will significantly increase the complexity of the system. Whether these additional features con tinues to improve the ability of this machine learning method to solve complex problems in human genetics still needs to be addressed.

Acknowledgments

This work was supported by National Institutes of Health (USA) grants LM009012, AI59694 and RR018787. We thank Dr. Wolfgang Banzhaf and the attendees of the 2007 Genetic Programming Theory and Practice (GPTP) Workshop for their insightful ideas about CE and complexity.

References

1. Bateson, W.: Mendel's Principles of Heredity. Cambridge University Press, Cambridge (1909)
2. Moore, J.H.: The ubiquitous nature of epistasis in determining susceptibility to common human diseases. Human Heredity 56, 73–82 (2003)
3. Moore, J.H., Williams, S.W.: Traversing the conceptual divide between biological and statistical epistasis: Systems biology and a more modern synthesis. BioEssays 27, 637–646 (2005)
4. Thornton-Wells, T.A., Moore, J.H., Haines, J.L.: Genetics, statistics and human disease: Analytical retooling for complexity. Trends in Genetics 20, 640–647 (2004)
5. Freitas, A.: Understanding the crucial role of attribute interactions. Artificial Intelligence Review 16, 177–199 (2001)
6. Li, W., Reich, J.: A complete enumeration and classification of two-locus disease models. Human Heredity 50, 334–349 (2000)
7. Lucek, P.R., Ott, J.: Neural network analysis of complex traits. Genetic Epidemiology 14, 1101–1106 (1997)
8. Ritchie, M.D., Hahn, L.W., Roodi, N., Bailey, L.R., Dupont, W.D., Parl, F.F., Moore, J.H.: Multifactor dimensionality reduction reveals high-order interactions among estrogen metabolism genes in sporadic breast cancer. American Journal of Human Genetics 69, 138–147 (2001)
9. Moore, J.H., White, B.C.: Genome-wide genetic analysis using genetic programming: The critical need for expert knowledge. In: Riolo, R.L., Soule, T., Worzel, B. (eds.) Genetic Programming Theory and Practice IV. Genetic and Evolutionary Computation, vol. 5, Springer, Heidelberg (2006)
10. Moore, J.H., White, B.C.: Exploiting expert knowledge in genetic programming for genome-wide genetic analysis. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 969–977. Springer, Heidelberg (2006)
11. Moore, J.H., Barney, N., Tsai, C.T., Chiang, F.T., Gui, J., White, B.C.: Symbolic modeling of epistasis. Human Heredity 63(2), 120–133 (2007)
12. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)

13. Koza, J.R.: Genetic Programming II: Automatic Discovery of Reusable Programs. MIT Press, Cambridge Massachusetts (1994)
14. Koza, J.R., Andre, D., Bennett, I.F.H., Keane, M.: Genetic Programming 3: Darwinian Invention and Problem Solving. Morgan Kaufmann, San Francisco (1999)
15. Koza, J.R., Keane, M.A., Streeter, M.J., Mydlowec, W., Yu, J., Lanza, G.: Genetic Programming IV: Routine Human-Competitive Machine Intelligence. Kluwer Academic Publishers, Dordrecht (2003)
16. Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D.: Genetic Programming – An Introduction. In: On the Automatic Evolution of Computer Programs and its Applications. Morgan Kaufmann, San Francisco (January, 1998)
17. Langdon, W.B.: Genetic Programming and Data Structures: Genetic Programming + Data Structures = Automatic Programming!, Genetic Programming, vol. 1. Kluwer, Boston (April 24, 1998)
18. Langdon, W.B., Poli, R.: Foundations of Genetic Programming. Springer, Heidelberg (2002)
19. Freitas, A.: Data Mining and Knowledge Discovery with Evolutionary Algorithms. Springer, Heidelberg (2002)
20. Fogel, G.B., Corne, D.W.: Evolutionary Computation in Bioinformatics. Kaufmann Publishers, San Francisco (2003)
21. Yu, T., Riolo, R.L., Worzel, B. (eds.): Genetic Programming Theory and Practice III. Genetic Programming, vol. 9. Ann Arbor, Springer, Heidelberg (May 12–14, 2005)
22. Banzhaf, W., Beslon, G., Christensen, S., Foster, J.A., Kepes, F., Lefort, V., Miller, J., Radman, M., Ramsden, J.J.: From artificial evolution to computational evolution: a research agenda. Nature Reviews Genetics 7, 729–735 (2006)
23. Moore, J.H., Parker, J.S., Hahn, L.W.: Symbolic discriminant analysis for mining gene expression patterns. In: Flach, P.A., De Raedt, L. (eds.) ECML 2001. LNCS (LNAI), vol. 2167, pp. 191–205. Springer, Heidelberg (2001)
24. Folino, G., Pizzuti, C., Spezzano, G.: A cellular genetic programming approach to classification. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999), pp. 1015–1020 (1999)
25. Greene, C.S., White, B.C., Moore, J.H.: An expert knowledge-guided mutation operator for genome-wide genetic analysis using genetic programming, vol. 4774, pp. 30–40 (2007)
26. Goldberg, D.E.: The Design of Innovation. Kluwer Academic Publishers, Dordrecht (2002)
27. Sempere, L.F., Cole, C.N., McPeek, M.A., Peterson, K.J.: The phylogenetic distribution of metazoan micrornas: insights into evolutionary complexity and constraint. Journal of Experimental Zoology 306, 575–575 (2006)